Chapter 3
Methodology

This chapter covers the methodology used to develop a solution for shortest path routing mentioned in the previous chapter. Two kinds of neural networks, which is Hopfield neural network and Boltzmann machine are chosen to perform neural computing algorithm for solving the shortest path problem. The Hopfield neural network model used in this research was based on the paper by Mustafa and Faouzi titled *"Neural Network for Shortest Path Computation and Routing in Computer Networks"* [28]. On the other hand, the Boltzmann machine is a modification made from the paper *"A Boltzmann Machine Solution of The Traveling Salesperson Problem: A Study For Parallel Implementation"* [29] by Tesar *et al*, which was originally proposed for solving the traveling salesperson problem.

3.1 Constructing a Neural Network for Solving COP

There are several steps in constructing a neural network for solving constraint optimization problem (COP) such as the shortest path routing problem. First, one should identify the objectives of the COP. The objectives of each specific COP instances are what the neural network is trying to solve. Thus, the objectives of a COP can be viewed as the constraints and requirements of the neural network.

The objectives of the COP are described mathematically in terms of the energy function of the neural network. One can map the objectives of the COP as the connection weights of the neural network, or the bias of a neuron or even the architecture of the neural network. For different kinds of neural network used, the objectives of the COP are mapped differently. After the problem is mapped into the neural network, the next step is to identify the values of other parameters. For example, the penalty parameters in the energy function of the Hopfield network, or the starting temperature of the Boltzmann machine.

6. Calculate the bias term.
7. Calculate the connection weights of each pair of neurons.
8. Repeat

   Update all neurons.
9. Until none of the neuron output voltage changes by more than the threshold value from one update to the next.
10. If neuron's final output value is greater or equal to 0.5 then set the neuron to ON. Else, set the neuron to OFF.

3.5 The Boltzmann Machine

This section focused on the proposed model of Boltzmann machine for solving the shortest path routing. Boltzmann machine is well known with its capability for solving constraint optimization problem such as the Traveling Salesperson Problem (TSP). The model proposed here is a modification from the Boltzmann machine proposed by Tesar *et. al.* in [29]. Modifications are made to the model's architecture, representation of neuron and connection weight to adapt this model to solve the shortest path routing problem.

In the study of [29], for the TSP with n cities, there are n^2 neurons. These neurons are arranged in a two-dimensional array, element U_{xi} in the array at row x represents the cities and column i represents the sequence of visiting the city. In other words, neurons in the same row represent the same city and the neurons in same column represent the time that the visit to the city is made. In the solution of TSP, since the salesperson cannot be in two cities at the same time, thus, there must be one and only one neuron in the "ON" state in each column or row.

3.3 The Representation of the Shortest Path Routing Problem

This section will focus on the representation of the shortest path routing problem in Hopfield neural network and the Boltzmann Machine used to determine the shortest path.

In the shortest path routing problem, each links in the computer network is assigned with a link cost. This link cost is usually described in the form of two-dimensional matrix. In this study, the link cost matrix for both the Hopfield network and the Boltzmann machine are represented by C. The element of C, C_{xi} is a finite real positive number which is the cost of the link between computer node x and computer node i. Another matrix is used to describe the connectivity of the computer network is represented by ρ where

$$\rho_{xi} = \begin{cases} 1 & \textit{if the arc from node x to node i does not exist} \\ 0 & \textit{otherwise.} \end{cases}$$

For non-existing link (ρ_{xi} = 1), the link cost is assigned with 0 (C_{xi}=0) in the Hopfield network but will be assigned with a very large number in the Boltzmann machine.

The Hopfield network in [28] and the proposed model of Boltzmann machine will be organized in an n x n matrix where n is the number of nodes in a computer network. Neurons in these two neural networks are represented by the links in the computer network. Output of a neuron is defined as:

$$V_{xi} = \begin{cases} 1 & \textit{if the arc from node x to node i is in the shortest path} \\ 0 & \textit{otherwise} \end{cases}$$

Table 3.1 shows the shortest path that connects source node 1 and destination node 6 which can be read as 1-2-5-6-1. In this research, the suggestion by Mustafa and Faouzi in [28] is used. The suggestion is to include the link that connects the destination node back to the source node although this link might not exist in the computer network. This is to avoid the problem of looping and to ensure that both source node and destination node are included in the solution. Since V_{xi} represents link connecting node x and node i, then V_{ii} represent the

self-connected link of a computer node. This kind of links should not be included in the solution. Avoiding the self-connected link can reduce the number of connections in the neural network and thus, simplify the neural network.

Table 3.1 Representation of the shortest path

	1	2	3	4	5	6
1		1	0	0	0	0
2	0		0	0	1	0
3	0	0		0	0	0
4	0	0	0		0	0
5	0	0	0	0		1
6	1	0	0	0	0	

3.4 The Hopfield Network

After the objectives and the representation are chosen, the next step is to define the energy function for the neural network. The energy function is derived with modification from the previous study. The energy function of the neural network describes the requirement and the constraints of the COP and the process of minimizing the energy function drive the neural network to a stable state.

The energy function for different kinds of neural network is different from each other. The energy function of a Hopfield network usually contains a several penalty parameters. Each penalty parameter expresses the preference of each of the constraint and requirement. In this research, the energy function of the Hopfield network is derived from [28] which is proposed by Mustafa and Faouzi.

This energy function is as follow:

$$E = \frac{\mu_1}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x}}^{n} C_{xi} \bullet V_{xi} + \frac{\mu_2}{2}\sum_{x=1}^{n}\sum_{\substack{i=1\\ i\neq x}}^{n} \rho_{xi} \bullet V_{xi} + \frac{\mu_3}{2}\sum_{x=1}^{n}\left\{\sum_{\substack{i=1\\ i\neq x}}^{n} V_{xi} - \sum_{\substack{i=1\\ i\neq x}}^{n} V_{ix}\right\}^2 + \frac{\mu_4}{2}\sum_{i=1}^{n}\sum_{\substack{x=1\\ x\neq i}}^{n} V_{xi} \bullet (1 - V_{xi}) + \frac{\mu_5}{2}(1 - V_{ds}) \tag{29}$$

$$(x,i) \neq (d,s) \qquad (x,i) \neq (d,s)$$

The formulation of this energy function is based on the objectives of the shortest path routing problem discussed in Section 3.2. The terms μ_1, μ_2, μ_3, and μ_4 are the penalty parameters in this energy function. Value chosen for these parameters must reflect the preferences of each constraint and requirement.

After the energy function of the Hopfield network is defined, one needs to obtain the equation of motion, that describes the minimization of the energy of the neural network. The original equations of motion for Hopfield network was introduce by J. J. Hopfield in [23]. The equation of motion is:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{30}$$

By substituting the energy function in equation (29), the equation of motion for the Hopfield network is obtained:

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\mu_1}{2} C_{xi}(1 - \delta_{xd} \cdot \delta_{is}) - \frac{\mu_2}{2} \rho_{xi}(1 - \delta_{xd} \cdot \delta_{is}) + \mu_3 \sum_{\substack{y=1\\ y\neq i}} (V_{iy} - V_{yi}) - \frac{\mu_4}{2}(1 - 2V_{xd}) + \frac{\mu_5}{2} \delta_{xd} \cdot \delta_{is} \tag{31}$$

$$\forall (x,i) \in \overline{Nx}\,\overline{N} \mid x \neq i$$

where δ is the Kronecker delta defined by

$$\delta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

The connection strength and the biases for energy function are

$$T_{xi,yi} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy} \qquad \textbf{Equation 32}$$

$$I_{xi} = -\frac{\mu_1}{2}C_{xi}(1-\delta_{xd}\cdot\delta_{is}) - \frac{\mu_2}{2}\rho_{xi}(1-\delta_{xd}\cdot\delta_{is}) + \mu_3\sum_{\substack{y=1\\ y\neq d}}(V_{iy}-V_{yi}) - \frac{\mu_4}{2}(1-2V_{xd}) + \frac{\mu_5}{2}\delta_{xd}\cdot\delta_{is}$$

$$= \begin{cases} \frac{\mu_5}{2} - \frac{\mu_4}{2} & if(x,i)=(d,s) \\ -\frac{\mu_1}{2}C_{xi} - \frac{\mu_2}{2}\rho_{xi} - \frac{\mu_4}{2} & otherwise \end{cases}$$

$$\forall(x \neq i), (y \neq i) \qquad \textbf{Equation 33}$$

The final step in building the Hopfield network is to set values for the coefficients in the energy function. Setting the value of the coefficient is a critical and controversial step because the value of the coefficients will affect the convergence of the network and there is no way to gather the exact value of the coefficients. When the value of each coefficient is determined, we need to simulate the Hopfield network. During the simulation, one can decide whether random noise needs to be added to the network or not. The act of adding noise to the network is to help it converge at the global minima.

3.4.1 Hopfield Model Algorithm for Solving Shortest Path Routing

The following procedure is used for solving the shortest path routing problem using a Hopfield network. It is proposed in [28]:

1. Get the cost and connection matrix of the computer network.
2. Setting values for the coefficients in the energy function.
3. Setting value for the gain parameter.
4. Setting initial input voltage for all neurons.
5. Add initial random noise to all neurons.

6. Calculate the bias term.
7. Calculate the connection weights of each pair of neurons.
8. Repeat

   Update all neurons.
9. Until none of the neuron output voltage changes by more than the threshold value from one update to the next.
10. If neuron's final output value is greater or equal to 0.5 then set the neuron to ON. Else, set the neuron to OFF.

3.5 The Boltzmann Machine

This section focused on the proposed model of Boltzmann machine for solving the shortest path routing. Boltzmann machine is well known with its capability for solving constraint optimization problem such as the Traveling Salesperson Problem (TSP). The model proposed here is a modification from the Boltzmann machine proposed by Tesar *et. al.* in [29]. Modifications are made to the model's architecture, representation of neuron and connection weight to adapt this model to solve the shortest path routing problem.

In the study of [29], for the TSP with n cities, there are n^2 neurons. These neurons are arranged in a two-dimensional array, element U_{xi} in the array at row x represents the cities and column i represents the sequence of visiting the city. In other words, neurons in the same row represent the same city and the neurons in same column represent the time that the visit to the city is made. In the solution of TSP, since the salesperson cannot be in two cities at the same time, thus, there must be one and only one neuron in the "ON" state in each column or row.

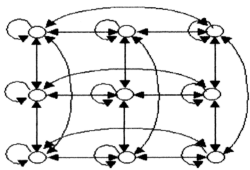

Figure 3.1 Boltzmann machine architecture

In the case of the shortest path routing, neuron U_{xi}, which is located at row x and column i is used to represent a computer node and the sequence of computer node in the shortest path by its row and column respectively. Neuron U_{xi} can also be used to represent the link connecting computer node x and computer node i. However, in the proposed model of Boltzmann machine, these neurons are used to represent the links in the computer network.

The next consideration of the Boltzmann machine for solving the shortest path routing is its architecture. Neurons in Boltzmann machine are fully connected and each neuron has a self-connected link. The self-connected link can be viewed as the bias of the neuron. Connection weights and the bias force a neuron to "ON" or "OFF" state. There are three types of connection weights in the Boltzmann machine for TSP:

1. Connection weight between neurons at the same row or same column.
2. Connection weight between neuron at different row and column.
3. Connection weight of a neuron back to itself (or the bias).

In the Boltzmann machine for solving the TSP, connection weight $w_{xi,xj}$ is the weight between two neurons U_{xi} and U_{xj} at row x. This kind of connection weight represent the constraint that each city should be visited once. The connection weight between neurons at the same column, $w_{xi,yi}$ represents the constraint that the salesperson cannot visit two cities at the same time. The connection weight between two neurons at the different row and different column $w_{xi,yj}$ is determined from the distance between city x and city y that the neurons represent.

This connection weight is assigned with a negative term because the shorter the distance between the two cities the neuron is more favorably to be ON. The connection weight of self-connected link for each neuron is the same because all the neurons equally prefer to be forced to ON state.

In the proposed model of Boltzmann machine for solving shortest routing problem in a n-node computer network, there are n^2 neurons in the Boltzmann machine. Each of the neuron in the Boltzmann machine represents the link between two computer nodes. For example, neuron U_{xi} represent a link between computer node x and computer node i. The link cost of the computer network is described using the cost matrix C, and the matrix ρ describes the connection of the computer network.

When the neurons are arranged into a two-dimensional matrix, the architecture of the proposed model of Boltzmann machine will be the same as in Figure 3.1. Neuron U_{xi} where $x = i$ refers to the self-connected link of computer node x. Self-connected link of a computer node does not usually exist in real computer network, thus, neuron U_{xi} where $x = i$ should be force to OFF state. Neurons located at the same row represent the computer links that originate from the same computer node. For instance, neuron U_{xi} and neuron U_{xj} that are located at row x represent the two links that are going out to computer node i and computer node j. Thus, the row index can also be used to refer a computer node. Hence, in the final solution for the shortest path problem, there should be at most one neuron ON for every row and column.

Since neurons in the proposed model of Boltzmann machine represent the link of the computer network, the link cost cannot be directly mapped into the connection weight as in the Boltzmann machine for TSP. The connection weight $T_{xi,yj}$ between two neurons at different row and different column ($x \neq y$ and $i \neq j$) is determined from the sum of the link cost of the two computer links that the neurons represent ($Link\_Cost_{xi} + LinkCost_{yj}$). The connection weight of the neurons located at the same row or the same column expresses the constraint that there must be at most one neuron ON for every row and column. The two kinds of connection weight discussed above are assigned with a negative term used for energy minimization.

The connections weight of a neuron back to itself also can be referred as bias. The bias is a positive number assigned to each of the neuron to describe the preference of the neuron to ON. In the proposed Boltzmann machine for shortest path routing, the bias is derived from the cost of the link that the neuron represents. The smaller the cost the neuron is more preferably to be ON. Therefore, the bias should be $Bias = \frac{1}{link \cos t}$.

For solving shortest path beginning with computer node x (source node) and ending with computer node i (destination node), the bias of all neurons that represent link from computer node x to other computer nodes should force to the neurons to ON state but only one of these neuron is allowed to be in ON state in the final solution. To ensure both source and destination node are in the shortest path, we follow the approach discussed in previous section which adds a link from destination node back to the source node, thus, the neuron that represents this link will always have a bias that force it to ON state.

In order to solve the shortest path routing problem, we also need to get the starting temperature of the Boltzmann machine and define the cooling schedule. The temperature should be cooled slowly. To fulfill this requirement, exponential cooling schedule is usually used. The exponential cooling schedule New_Temperature = α (Old_Temperature), reduces the temperature each time the neuron is updated. The value of α is usually in the range between 0.9 and 0.99. The value of α and starting temperature are often based on previous experimental result.

After choosing the starting temperature and the value for α, we have to define the stopping conditions. The stopping conditions are:

1. At most one neuron is ON for every row.
2. The row that represents the links that originate from the source node will have one and only one neuron ON.
3. Neuron that represents the link from destination node back to the source node is ON.

3.5.1 Boltzmann Machine Algorithm for Solving Shortest Path Routing

The following is the description for the Boltmann Machine algorithm:

1. Get the cost and connection matrix of the computer network.
2. Initialize weights to represent the constraints of the shortest path routing.
3. Setting value for starting temperature and the control parameter.
4. Setting initial input value for all neurons.
5. Repeat
   Calculate activation change of the neuron.
   Compute probability of acceptance of the change.
   Determine acceptance or rejection of the change.
   Reduce the control parameter
6. Until the stopping conditions are fulfilled.

3.6 Chapter Summary

This chapter has discussed the algorithm of the Hopfield neural network and the algorithm of Boltzmann machine for the shortest path application. The Hopfield neural network is not a novel idea; it is taken from paper [28]. The Boltzmann machine is a modification from paper [29] that was originally used for the TSP application. The derivation of connection weights and biases is also discussed in this chapter. In the next chapter, the simulation model and the results gathered from the simulation will be discussed.